TERMA
ALLIES IN INNOVATION

DEVELOPMENT OF NEW SYSTEMC IP MODELS AND OCP-IP SOCKETS

Luca Fossati, Technical Officer
Mariarosaria Cardone, Project Manager
Mattias Holm, Technical Project Manager
Ignacio López, System Administrator
Alberto Ferrazzi, Junior Software Engineer

Outline

- **Introduction**
  - Purpose of the project
  - SoCRocket overview
  - SystemC
  - TLM
- **Project**
  - Model development and extension
    - L2C
    - MSDRAM Controller
    - GRIOMMU
    - GRSPW2
    - IRQ(A)MP
  - Demonstrator platform
- **Conclusion**
  - Problems encountered
  - Achievements and Performances
  - What’s next

TERMA
ALLIES IN INNOVATION

INTRODUCTION

Project: purpose

- **Purposes of the project:**
  - Extend SoCRocket with new models to be able to simulate the NGMP:
    - L2C
    - GRIOMMU
    - GRSPW2
    - MSDRAM Controller
    - IRQ(A)MP
  - Demonstrate the functionalities of the new models in a virtual platform
  - Give feedbacks, improvement suggestions and guidelines for modelling with SoCRocket
- **The Next Generation Microprocessor (NGMP) is a microprocessor development project initiated by ESA for usage in the future missions.**
  - AMBA bus
    - One 128-bit Processor AHB bus
    - One 128-bit Memory AHB bus
    - Two 32-bit I/O AHB buses
    - One APB bus (register area)
  - Quad-core Leon 4 processor
  - More information at: http://microelectronics.esa.int/ngmp/ngmp.htm

Introduction: SoCRocket overview

- **Initially developed by TU Braunschweig in agreement with ESTEC**
- **It is a framework to assemble custom simulators of SoC (System on Chip) typically used in space.**
  - A simulator is created by configuring and connecting different models together.
  - Each model brings the functionalities of its hardware counterpart.
- **SoCRocket Purposes**
  - Early software development
  - Architecture exploration
- **Usage in the hardware design process**
  - Early stages, for preliminary verification, before VHDL production/usage.
  - VHDL stage to have a reference during modelling.
- **SoCRocket is composed by**
  - Models library (i.e. MSDRAM, L2C, ...)
  - Base classes that provides the principles for interconnecting the models
  - Configuration generator wizard
  - Build/Test execution system

Introduction: SystemC

- **Core Library that moves SoCRocket**
- **Allows to use C++ features to write “executable specifications” (C++ program that exhibits the same behaviour of the emulated system)**
- **Provides**
  - Scheduler: runs the simulation
  - Class library that provides the basic blocks to model any kind of system:
    - Models: partitioning of code
    - Processes: implement logic of model
    - Ports: pass data through processes
    - Signals: connect ports

- **Supports different level of abstraction from RTL (Register Transfer Level) to Functional**

Introduction: SystemC example

The following two pieces of code show the basics of SystemC:

- **Declare SystemC module**
- **Instantiate and connect modules together.**

```
3   //·Declare·a·module·that·performs·an·AND·operation.
4   SC_MODULE(AndModule)
5   {
6   ··//·Declare·input·and·output·ports.
7   ··sc_in<bool>·inA,·inB;
8   ··sc_out<bool>·outV;
9
10  ··//·This·function·expresses·the·logic·of·the·model.
11  ··void·do_and()
12  ··{
13  ····outV.write(·inA.read()·&&·inB.read()·);
14  ··}
15
16  ··SC_CTOR(AndModule)
17  ··{
18  ····//·Register·do_and·method·to·be·executed·when·there·is
19  ····//·a·change·of·value·in·ports·inA·or·inB.
20  ····SC_METHOD(do_and);
21  ····sensitive·<<·inA·<<·inB;
22  ··}
23  };
```

```
27  ··//·Declare·modules·and·signals.
28  ··AndModule·moduleA(sc_module_name("ModuleA")),
29  ··········moduleB(sc_module_name("ModuleB"));
30  ··sc_signal<bool>·sigA;
31
32  ··//·Ports·binding.
33  ··moduleA.outV(sigA);
34  ··moduleB.inB(sigA);
```

Introduction: TLM2.0

- **TLM = Transaction Level Modelling**
  - Focus is on modelling the transactions on the bus
  - Transactions are modelled as calls to a function
    - Example: ahb.b_transport(dataPayload, delay);
  - Just a set of standard interfaces that have to be implemented by the models
  - Faster then RTL
- **The support to this kind of modelling is provided to SystemC by TLM library.**
- **Two coding style that use 2 different transport interfaces**
  - LT – Transaction complete with the return of a blocking call. Models are allowed to run ahead of simulation time. Faster but less accurate simulation. Used for software development.
  - AT – Transaction is modelled with a set of non blocking calls. This allow modelling of phases of the (bus) protocol. Models remains synchronized with simulation time. Better timing accuracy but slower simulation. Used for architecture exploration purpose.
- **SoCRocket models must support both these code styles.**
- **SoCRocket provides some base classes that abstract the code styles so the model developer does not have to deal with them in simple cases.**

Introduction: TLM example

```
24  void MemoryModel::b_transport_implementation(tlm_generic_payload &trans,
25  ··sc_core::sc_time &delay)
26  {
27  ··//·Get·the·transaction·address
28  ··uint64_t·regAddr·=·trans.get_address();
29  ··if·(registers.fallOnRegisterArea(regAddr))
30  ··{
31  ····registers.write(regAddr,·trans.get_data_ptr());·//·Perform·the·action.
32  ····trans.set_response_status(TLM_OK_RESPONSE);
33  ··}
34  ··else
35  ··{
36  ····trans.set_response_status(TLM_ADDRESS_ERROR_RESPONSE);
37  ··}
38
39  ··delay·+=·1·*·clockCycleTime;··//·Add·delay·due·to·this·operation.
40  }
```

Implement the transport function (LT).

```
43  SC_MODULE(MemoryModel)
44  {
45  public:
46  ··simple_target_socket<MemoryModel>·targetSocket;·//·Declare·a·TLM·socket.
47
48  ··SC_CTOR(MemoryModel)·:·clockCycleTime(10,·sc_time_unit::SC_NS)
49  ··{
50  ····targetSocket.register_b_transport(this,·&MemoryModel::b_transport_implementation);
51  ··}
52
53  ··void·b_transport_implementation(tlm_generic_payload·&·trans,·sc_core::sc_time·&·delay);
54
55  ··sc_time·clockCycleTime;
56
57  ··Registers·registers;
58  };
```

Declare a target socket and register the transport function (b_transport)

```
71  ··initiatorModel.AHB(memoryModel.targetSocket);
```

Bind target and initiator sockets

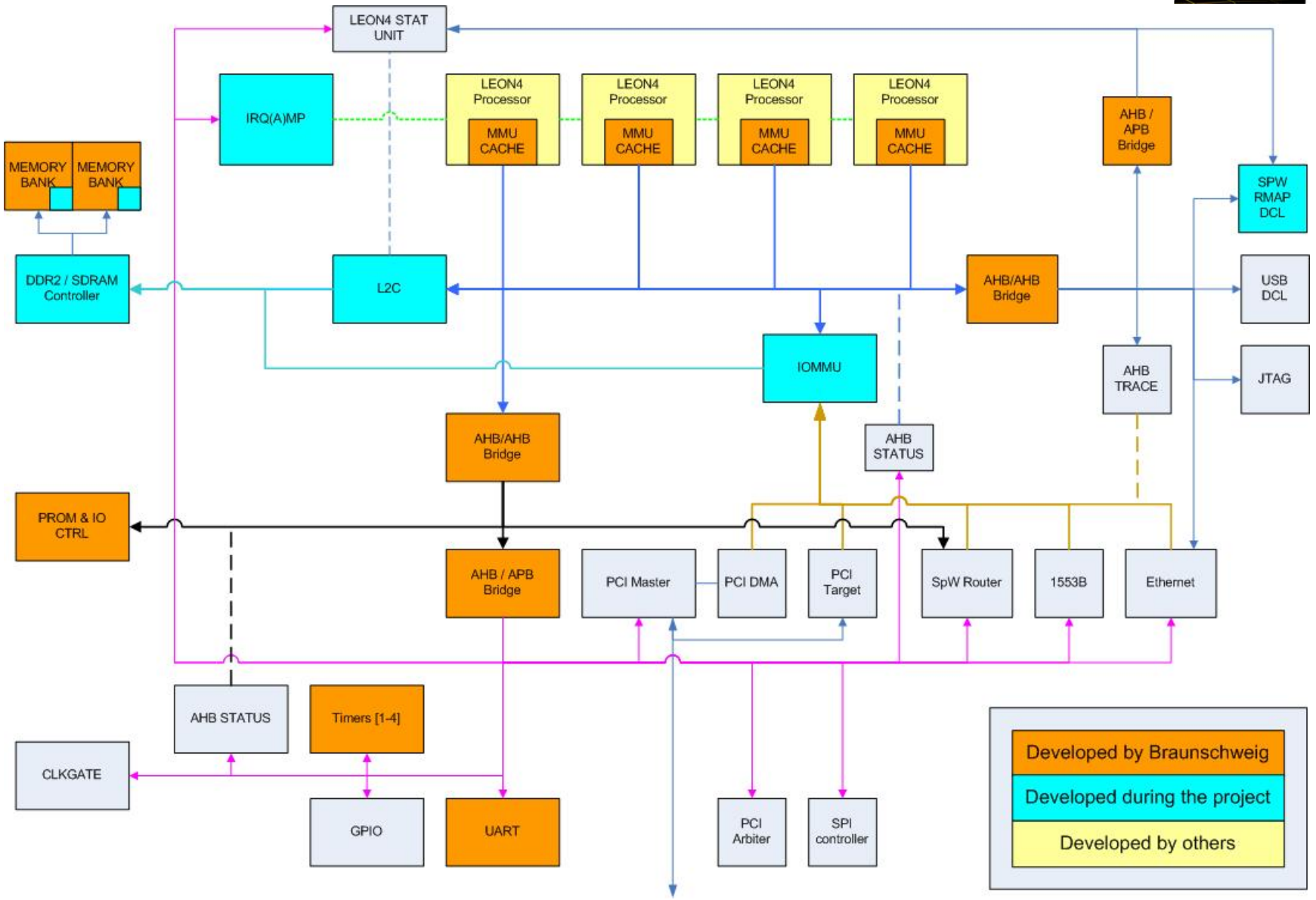
TERMA
ALLIES IN INNOVATION

PROJECT

Project: Overview

- **Overview of the models developed**
  - L2C
  - MSDRAM Controller
  - GRIOMMU
  - IRQ(A)MP
  - GRSPW2
- **Virtual Platform demonstrator**
  - Demonstrator
  - Demonstrator programs
  - Run the demonstrator

Overview of the models developed

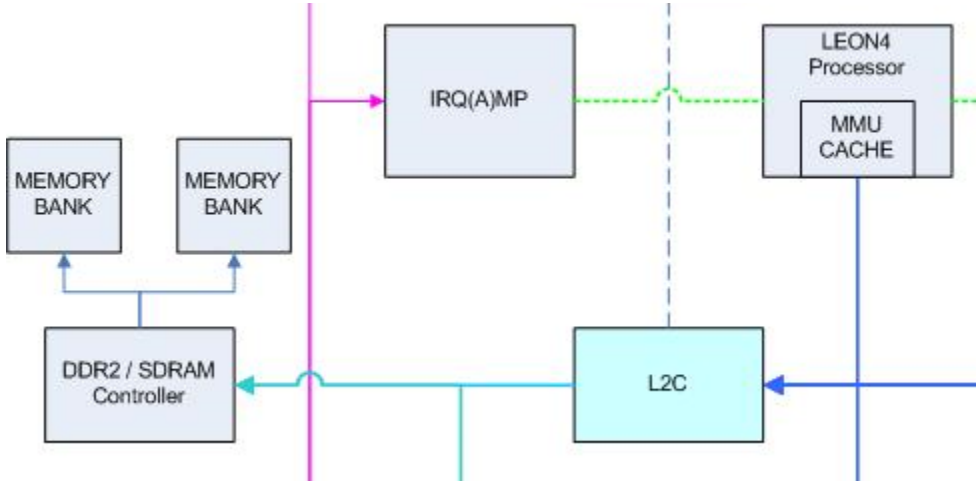


Model developing approach

- **Read model documentation**
  - Theoretically model documentation is enough to build good models, as we are doing behavioral modelling and the documentation indicates:
    - The functionalities of the model, describing the effects that can be seen over its interfaces.
    - The timing aspect of each functionality
  - Practically a prototype board was required to solve inconsistencies or data omitted in the documentation (i.e. initial values of some registers).
- **Create the list of requirements, define the testing plan, and agree on it with ESTEC.**
- **Develop the model**
- **Develop test-benches to test the requirements**
- **Models integration into a virtual platform to test they works together**

L2C

- **This model implement the Level 2 Cache from GRLIB.**
- **Its purpose is to speed up operations on memory by caching.**
- **The speed up relies on the principle of locality**
  - Software accesses to memory usually presents two type of locality:
    - Temporal: same data is accessed multiple time
    - Spatial: sequential accesses happen usually on data located nearby.
- **To exploit this principle, when the L2C receives a request, it loads a full contiguous block of memory containing also the requested data in its own faster but smaller memory.**
- **Subsequent accesses to the requested data (temporal locality) or to near data (spatial locality) will result in faster accesses as the data is served from the faster cache memory.**



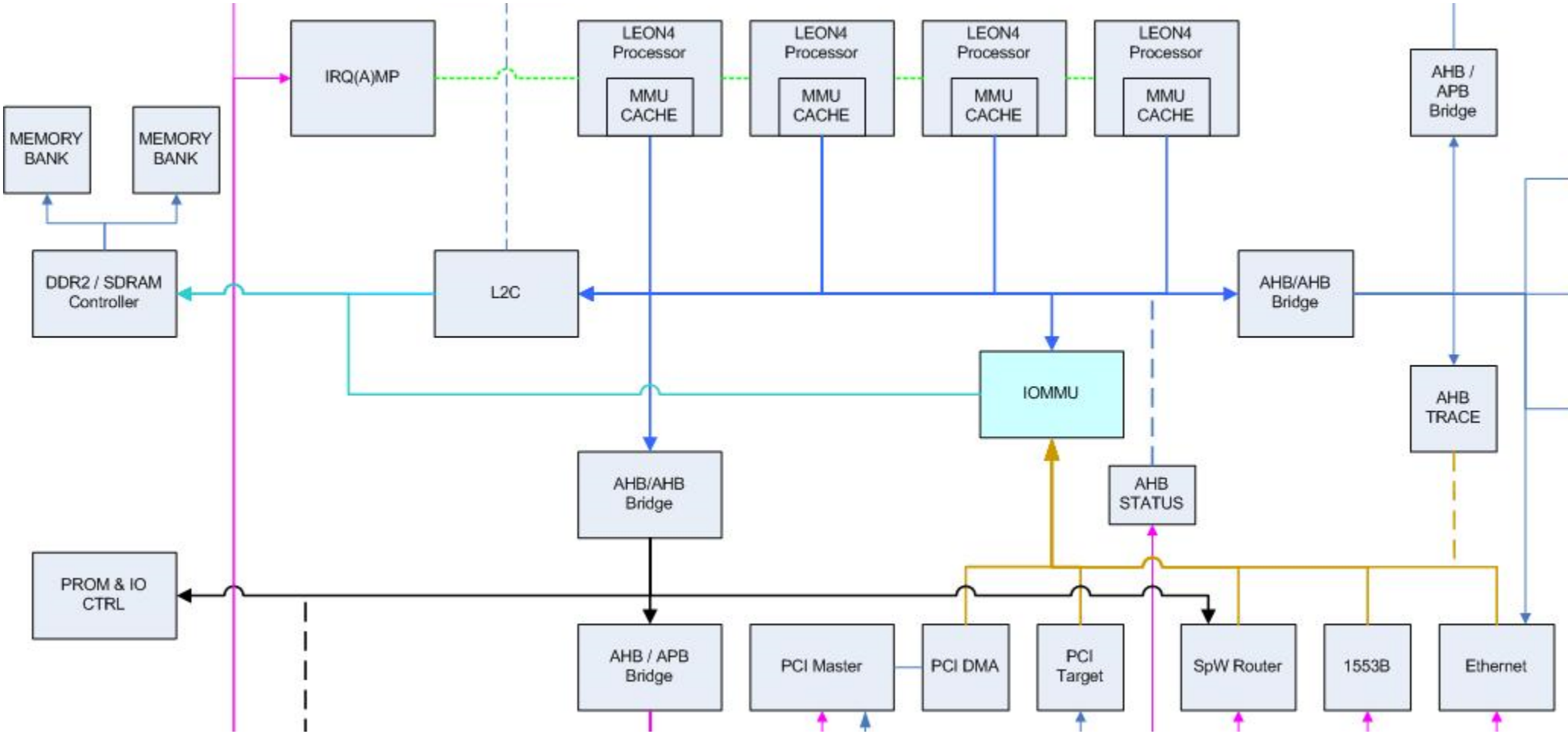
L2C: Features

- **Write Policies**
  - Write-through: write to the memory is queued and happen as soon as possible
  - Write-back: write to the memory happens when the cache line is replaced
- **Replacement Policies**
  - LRU (least recently used)
  - Pseudo-Random
  - Master-Index
- **Memory Type Range Register (MTRR)**
  - Each register allows to modify caching behavior on an address range
    - Write through
    - Not cacheable
    - Write protection
  - Up to 32 MTRR
- **Scrubber**
  - Prevent the accumulation of errors in the cache by checking each cache line and correcting the errors.
- **Diagnostic Interface access cache data and EDAC**

L2C: Features

- **Way locking**
  - Lock some ways so that they cannot be replaced
- **HPROT signal support**
  - HPROT signal can be used to override cacheablity
- **Support two type of cache**
  - Direct-Mapped
    - An address can be stored in only one cache line.
  - Set-Associative
    - An address can be stored in more then one cache line. The cache lines in which an address can be stored are called “ways”.
    - 2,3,4 Ways.
- **Size and cache line size configurable**
- **C++ interface to get statistics**

GRIOMMU



GRIOMMU: Main functionalities

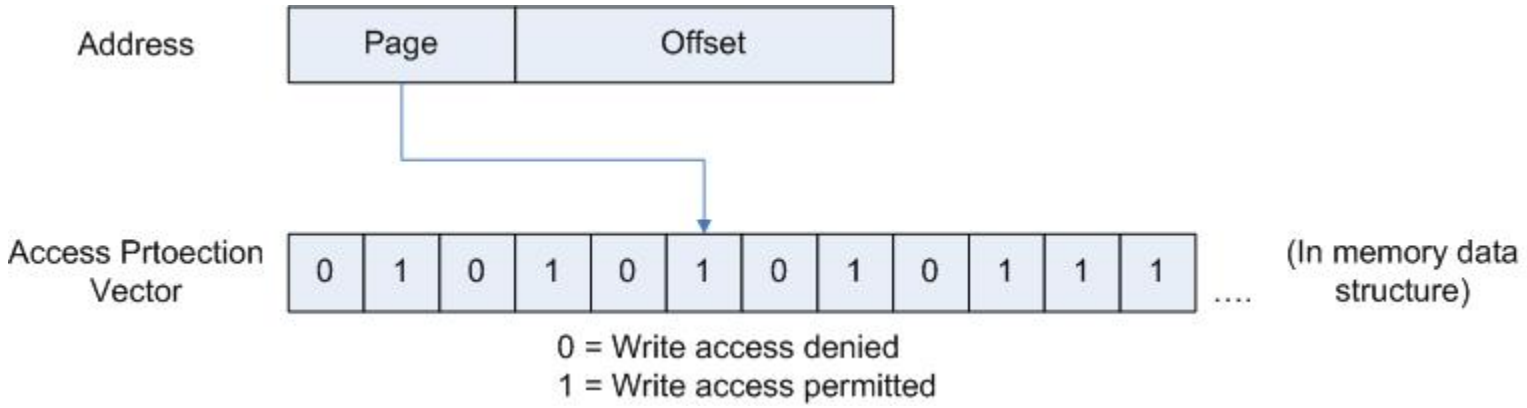
- **It is a bridge between two AHB busses**
- **Two main functionalities**
  - Access protection
    - Protect the memory from unwanted accesses by devices capable of Direct Memory Access (i.e. SpaceWire, Milbus, Can)
  - Address Translation
    - Access to a physical address is redirected to other physical addresses.

GRIOMMU: Access protection models

- **The device can be implemented to provide support for two model of access protection:**
  - Access Protection Vector (APV)
  - IOMMU
- **Both models rely on data structures in memory that define access protection and address translation.**
- **Memory is organized in pages, contiguous blocks of memory. Access protection and address translation is specified per memory page.**

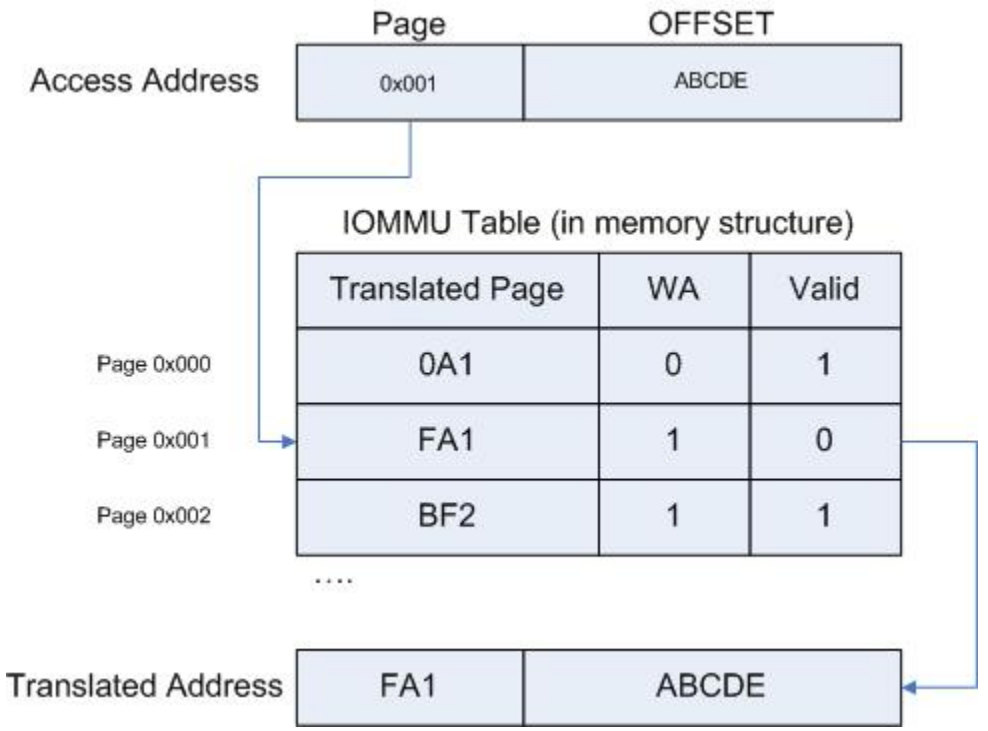
GRIOMMU: Access Protection Vector (APV)

- **Supports memory protection but not address translation**
- **The data structure used by APV is a simple bit vector where each bit value specifies the permission to write to the related memory page.**



GRIOMMU: IOMMU

- **Supports memory protection and address translation**
- **The IOMMU data structure is a table where each row defines access properties for the related page of memory.**
- **Each row specifies:**
  - Page address for address translation (used redirect the access to another memory page).
  - Write allowed (used to inhibit write access to the page)
  - Valid (specify if the row is valid)

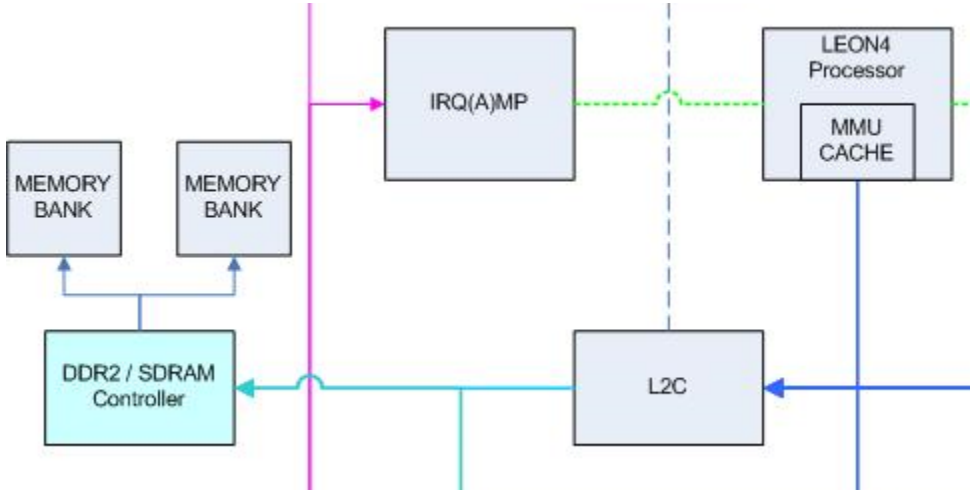


GRIOMMU: Features

- **Grouping**
  - Each master can be assigned to a group. Each group can have its own data structure that defines access protection / address translation.
- **Cache**
  - The GRIOMMU can be configured to use a direct mapped cache for accessing the data structure.
  - Is fault tolerant as it implements parity bit error detection. An error is treated as a cache miss.
  - Has a Diagnostic interface that allows to access cache data and inject errors.
- **ASMP**
  - Allows separate instances of software to control different masters without interfering with each others.
  - The register interface is mirrored on 4KIB boundary. Each mirrored interface can be configured to prevent write accesses to some registers (i.e. group register 1 can be written only from ASMP interface 1 and group register 2 only from two).

MSDRAM Controller

- **This model implements the DDRMUXCTRL device of the NGMP. It's not a GRLIB device.**
- **Device functionality: manage the access to SDRAM memory banks. This is a complex task that involves issuing different commands at right time.**
- **Support 2 different back-ends (NGMP baseline):**
  - PC-133 SDRAM
  - DDR2-800 (DDR are SDRAM with double pumping of data at both high and low edge of the clock)



MSDRAM Controller

- **Register set depends on back-end chosen but FT registers are common**
- **Timing is based on the values configured in registers. DDR timing parameters:**
  - T_{RP} = time between pre-charge and activate commands.
  - T_{RFC} = auto refresh period
  - T_{RCD} = time between activation and read/write commands
  - T_{RC} = time between two active commands
  - T_{RAS} = time required between active and pre-charge commands.

MSDRAM Controller: Features

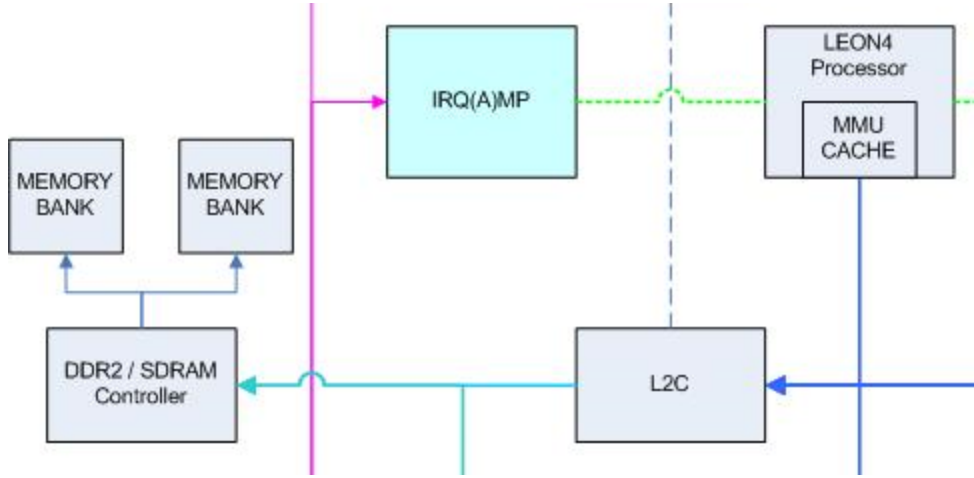
- **Configurable internal-bus size:**
  - Half-width: 32 bit
  - Full-width: 64 bit
- **EDAC (fault tolerance)**
  - Based on Reed-Solomon algorithm
  - Can be enabled/disabled. Bits are always written. Disabling save read/correction time.
  - Two modes: A, B

MSDRAM Controller: Implementation

- **SoCRocket improvement:**
  - Memory bank models have been improved to account also for read and write command delay times.
  - An external library that provides the Reed-Solomon error correction algorithm has been integrated into SoCRocket.

IRQ(A)MP

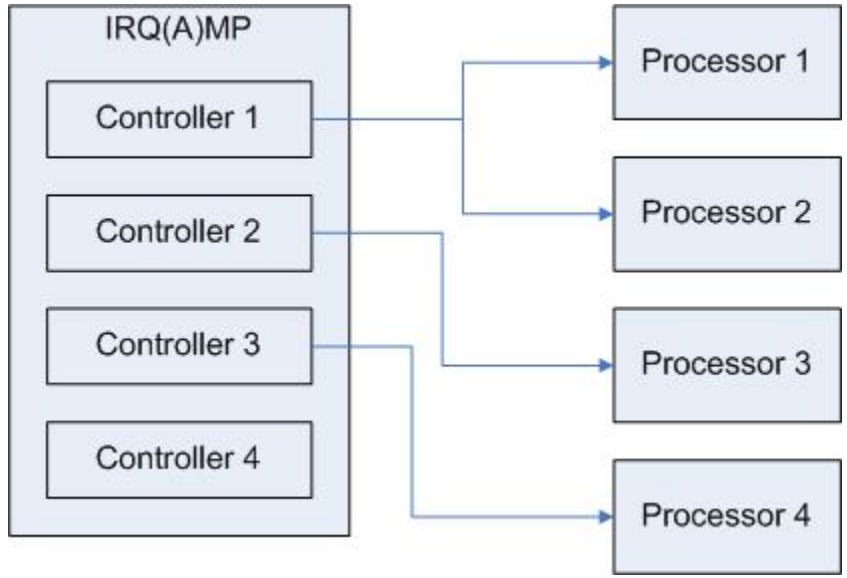
- **An Interrupt controller is a device that collects interrupts from different sources and forwards them to processor handling:**
  - Prioritization
  - Masking
  - Propagation
- **IRQ(A)MP is an interrupt controller that supports**
  - Asymmetric multiprocessing: each processor must be able to configure the interrupts.



IRQ(A)MP

IRQ(A)MP duplicates part of the interrupt controller circuitry and control registers to allow different interrupts handling for each processor.

Processors can be connected to different internal controllers for asymmetric multiprocessing or to the same for symmetric multiprocessing.



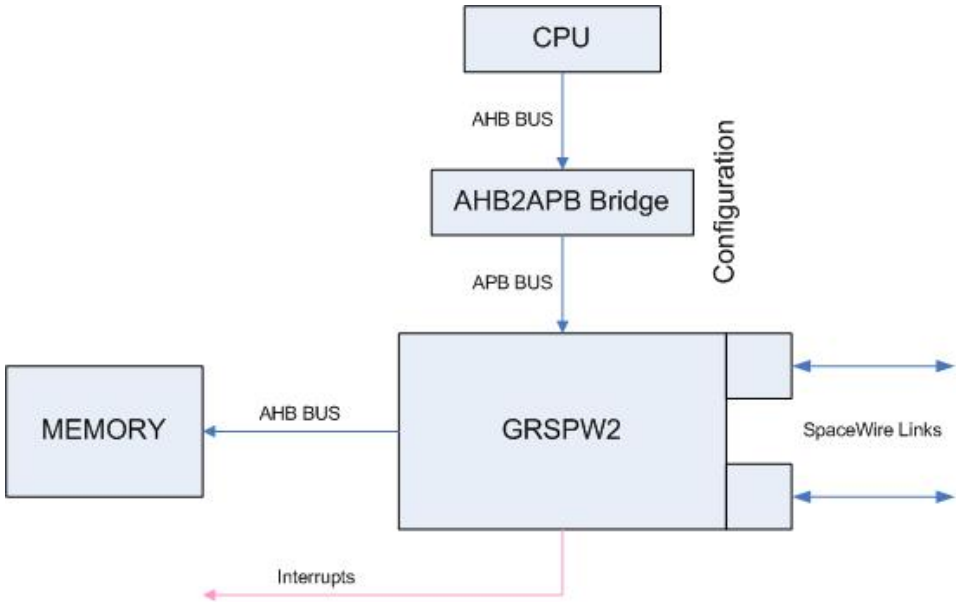
IRQ(A)MP

- **The implementation is based on the old IRQMP controller. On top we added the following features:**
  - Support for Asymmetric multi-processing
    - IRQ(A)MP can be configured with up to 16 internal interrupt controllers.
  - Watchdog
    - Used to assert a bit in the interrupt pending register when an external watchdog signal is asserted
  - Dynamic reset
    - Allows each processor to be booted from different specified memory addresses.
  - Time stamping
    - Allows to calculate the time used by the processor to execute a trap.
  - Broadcasting
    - The interrupt is sent to all processors and all processors must acknowledge it.

GRSPW2

- **This device connects the SoC to a SpaceWire link**
- **Interfaces**
  - Register interface on APB bus
  - Access memory through AHB bus
  - 2 SpaceWire ports
    - Character Interface

- **Functionalities:**
  - **Send and receive SpaceWire packets**
  - **Remote Memory Access Protocol (RMAP)**



GRSPW2: DMA Engine

The model uses DMA (Direct Memory Access) to store received packets in memory or fetch packets that have to be sent.

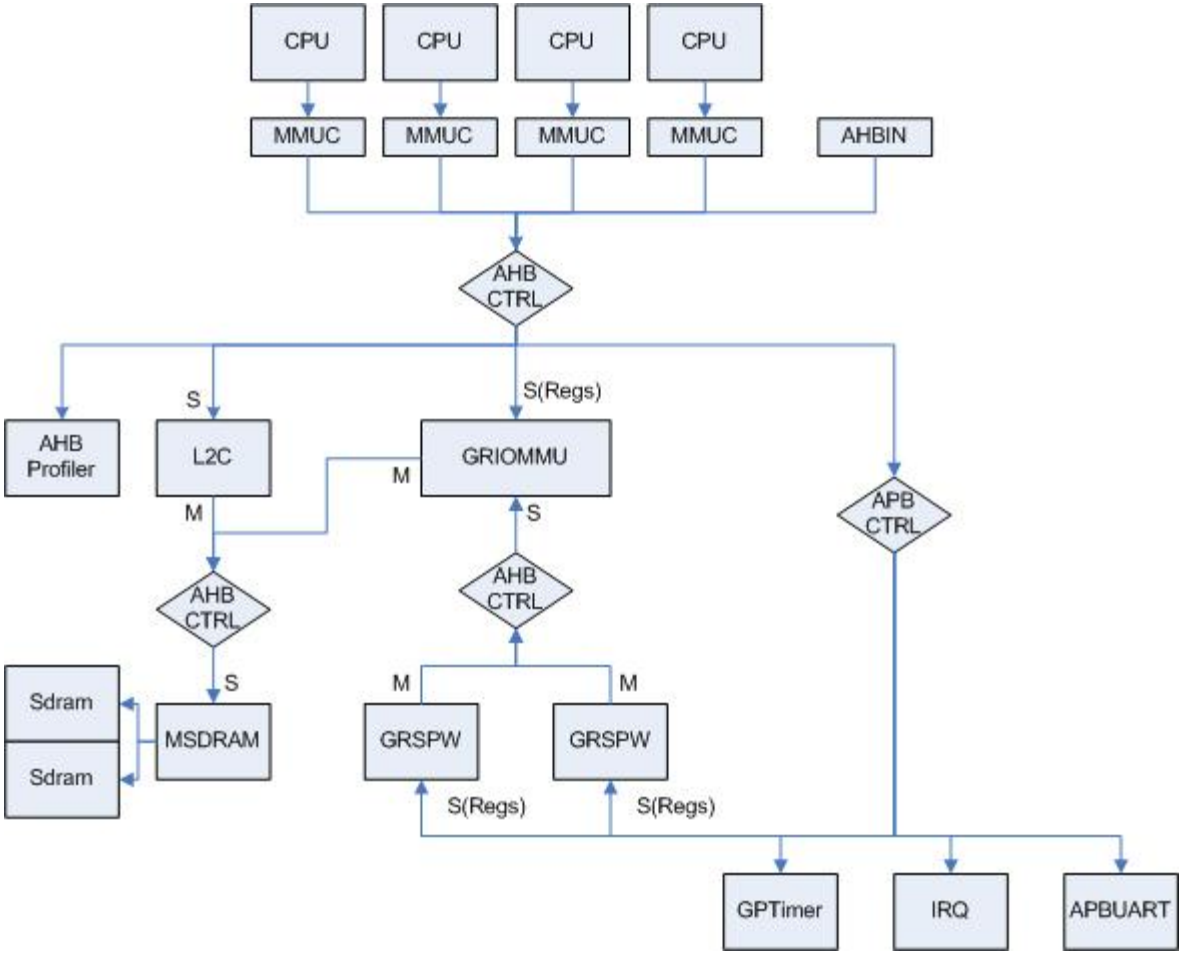
- **DMA operations are performed by DMA engines**
- **Up to 4 DMA engines**
- **Each DMA engine can be configured to receive or send different packets.**
- **Engine configuration resides in memory like the packet data as a list of “descriptors”.**

GRSPW2: RMAP

- **The device implements also the Remote Memory Access Protocol (RMAP)**
- **This protocol allows accesses to the resources reachable on the AHB bus from the SpaceWire link (i.e. a register value can be read or set from the SpaceWire)**
- **Runs in background (no need of actions from software)**

Demonstrator

- **In order to demonstrate the usage of the new developed components we assembled a virtual platform.**
- **The architecture is as close as possible to NGMP.**
- **Customizable via JSON configuration file.**
- **The XML template for the configuration wizard has been developed as well so the wizard can be used to produce the JSON configuration file.**



Demonstrator

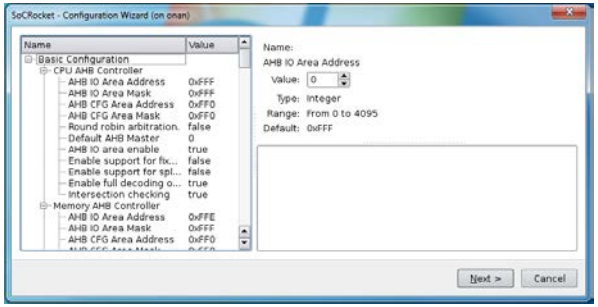
The demonstrator can run programs compiled with BCC SPARC compiler.

In order to run a program the user needs:

- Boot code image
- Program image

The user can optionally provide

- JSON configuration file
  - Overwrites the default configuration parameters for the models



Leon4mp.platform

Configuration.json

l4mp.bootcode
L4SysInfo.sparc

Demonstrator programs

- **The following programs have been written targeting the demonstrator platform to test the basic functionalities of the developed components:**
  - Boot code
    - Assembler program that setup the platform to run a c test programs.
  - Test programs: c programs that test the main functionality of the models.
    - GRIOMMU test
    - GRSPW2 test
    - L4SysInfo: plug and play discovery and information on developed models.

  Note: L2C, IRQ(A)MP and MSDRAM are tested implicitly by GRIOMMU and GRSPW2 tests.

CONCLUSIONS

Conclusion - Delivered Items

- **Documents**
  - Model user & developer manual
  - Models verification plan
  - SoCRocket developing guidelines
- **Software**
  - Models code
    - C++ code of each model
    - Test benches
  - Demonstrator platform code
    - Template for the configuration wizard
    - C++ code of the platform
  - Sample code demonstrating usage of the newly developed components

Conclusion – Problems encountered

- **Bus size**
  - The entire infrastructure is developed with 32 bit bus only in mind.
    - The AHBMaster and AHBSlave base classes come with a socket hardcoded to 32 bit (through template)
    - Templated sockets are a vantage and a disadvantage
      - Vantage: binding compatibility at compile time
      - Disadvantage: add bus configurability to a model requires to template it ➔ everything become a template
    - AHBCtrl, AHBMaster, AHBSlave functions are hardcoded for 32 bit bus
- **Split**
  - Split phase is not supported by the AHBCtrl
  - The base classes provided to develop the models do not allow any control over AT phases
- **Bugs & limitations**
  - GreenRegs do not allow direct read prevention to a register
  - AHBCtrl stuck due to a not initialized variable (took a while to find)
- **Code for AT protocol is quite complex multithread code**
- **Hardware models cannot be written with pure functions ➔ complex to write/test code**
- **Lack of documentation**
  - No guidelines for modelling with SoCRocket
  - No solutions for multi-bus components

Conclusion – Problems encountered

- **EDAC algorithms**
  - Hard to find open source libraries that implement them

Conclusion – Achievements

- **Platform performance**
  - Disclaimer: no results to show for comparison
  - Measured on Xeon 8-core processor with SpecInt = 43
    - Average transactions/sec per model ~ 45000/sec
    - Transactions/sec on main bus: 5000-12555
    - Performance are higher than the ones that may be obtained from a VHDL system.
      - This makes preliminary research and design of new architecture faster and cheaper.
      - Allow the platform to be used for software development
- **The core part of NGMP can now be simulated as all the device attached to CPU and memory bus are implemented. Only secondary device like Milbus or LAN controller are missing.**
- **Produced a document containing all the experience we gained so far in writing SoCRocket models.**

Conclusion – What's next

- **Improving SocRocket**
  - Bus size configurability
  - Split functionality
  - Validation against reference models (hardware)
  - Code improvements for maintainability
- **Extending SoCRocket**
  - Secondary models to complete simulation of NGMP SoC and models to support other architectures
- **Future work**
  - Bus-size configurability and split for of basic models
  - Validation
  - New models

Meet us at…

www.terma.com

www.terma.dk/press/newsletter

www.linkedin.com/company/terma-a-s

www.twitter.com/terma_global

www.youtube.com/user/TermaTV

TERMA

ALLIES IN INNOVATION